Titel: Android App-utveckling - Tekniskt PM
Namn: Love Jansson
ID: lovja643

1. Introduktion

Jag utvecklar en film-app som ska lista filmer och serier inom kategorin lgbtq. Varje film och serie har en detaljsida med mer information t.ex. trailer och bilder. Det är också möjligt för en inloggad användare att spara filmer och serier i egna samlingar.

Innan jag påbörjade appen samt under utvecklingens gång har jag försökt ta reda på hur en app i android kan byggas för att gör det lättare för mig att testa koden och underlätta eventuell utökning av applikationen senare när jag har mer erfarenhet. Eftersom appen i huvudsak ska hämta information från ett API är det även viktigt att detta hanteras på ett sätt som minimerar väntetiden för användaren.

I detta tekniska PM beskriver jag tre områden angående en Android-apps arkitektur som jag har läst om och sedan använt mig av eller förhållit mig till i projektet.

2. Design Patterns

Det verkar finnas en delad åsikt om att tillämpning av ett designmönster (eng. design pattern) kan öka en applikations kvalité [1][2]. De designmönster som framförallt har nämnts i samband med utveckling av Android-applikationer har varit MVC, MVP och MVVM. När jag har läst om dessa har jag fått uppfattningen av att MVC är det “ursprungliga” designmönstret som MVP och MVVM sedan har byggt vidare på i syfte att förbättra designen. [3] visar till exempel att många etablerade applikationer har tillämpat MVC samtidigt som [1] tyder på att MVP och MVVM föredras av utvecklare idag och [4] visar att dessa kan öka kodens testbarhet, modifierbarhet och prestanda i jämförelse med MVC. Android har också publicerat en guide [5] som bygger på MVVM. Denna guide utgår ifrån att applikationen hämtar data från nätverket och visar detta för användaren på något sätt, och detta var precis vad jag skulle göra varav jag valt att följa deras rekommendation.

3. Dependency Injection

Sedan tidigare är jag införstådd i att väl avgränsade funktioner och klasser underlättar vid test och hantering av koden. Något nytt som jag lärt mig om i detta projekt och som ytterligare delar in koden i olika ansvarsområden, är användning av ett ramverk för dependency injection. Jag läste att detta ska göra det betydligt enklare att hantera dependencies samt

underlätta vid testning av koden då möjligheten att införa mocks istället för riktiga dependencies till den testade enheten möjliggörs [1]. I och med detta valde jag att använda ett ramverk för dependency injection som heter Dagger. Nu när jag har börjat att skriva tester för koden kan jag hålla med om att användningen av Dagger förmodligen har gjort det enklare för mig. Mest för att de flesta dependencies tillförs klasser via konstruktorn varav jag kunnat göra Mocks och ge dessa som argument vid testning. Innan jag läste om dependency injection tänkte jag inte på detta utan jag instansierade t.ex. min Repository-klass inuti konstruktorerna för Viewmodels.

4. Data

En viktig del av applikation var som tidigare nämnt, att hämtning av data från nätverket inte skulle innebära några längre väntetider för användaren. I både [1] och i Android's guide framgår att lagring av data temporärt lokalt är något som förbättrar detta eftersom att nätverks request inte behöver ske lika ofta. I bägge källor rekommenderas användning av Android's Room vilket är ett lager ovanpå SQLite som ska underlätta hanteringen av databasen t.ex. genom annoteringar som genererar SQL queries. Eftersom jag inte har egen erfarenhet inom området valde jag att använda Room då det rekommenderas och ingår i tutorials samt guider.

Referenser

[1] Verdecchia, R., Malavolta, I., Lago, Patricia. (2019). Guidelines for Architecting Android Apps: A Mixed-Method Empirical Study. *IEEE International Conference on Software Architecture (ICSA)*, Hamburg, Germany, 2019, pp. 141-150, doi: 10.1109/ICSA.2019.00023.

[2] Carvalho, S.G., Aniche, M., Veríssimo, J. *et al.* An empirical catalog of code smells for the presentation layer of Android apps. *Empir Software Eng* 24, 3546–3586 (2019). https://doi.org/10.1007/s10664-019-09768-9

[3] Campos, E., Kulesza, U., Coelho, R., Bonifácio, R. and Mariano, L. (2015). Unveiling the Architecture and Design of Android Applications - An Exploratory Study. *In Proceedings of the 17th International Conference on Enterprise Information Systems - Volume 1: ICEIS,* ISBN 978-989-758-097-0, pages 201-211. DOI: 10.5220/0005398902010211

[4] Lou, T. A comparison of Android Native App Architecture: MVC, MVP and MVVM. Master thesis, Aalto University, 2016. Available from: https://research.tue.nl/en/studentTheses/a-comparison-of-android-native-app-architecture

Titel: Android App-utveckling - Tekniskt PM
Namn: Love Jansson
ID: lovja643

[5] Android’s guide to app architecture: https://developer.android.com/jetpack/guide